DOCKER CONTAINER UND VIRTUALISIERUNG

AM ENDE DIESES MODULS ...

... verstehen Sie den Unterschied zwischen einem Container und einer virtuellen Maschine.

... kennen Sie die Vorteile eines Containers.

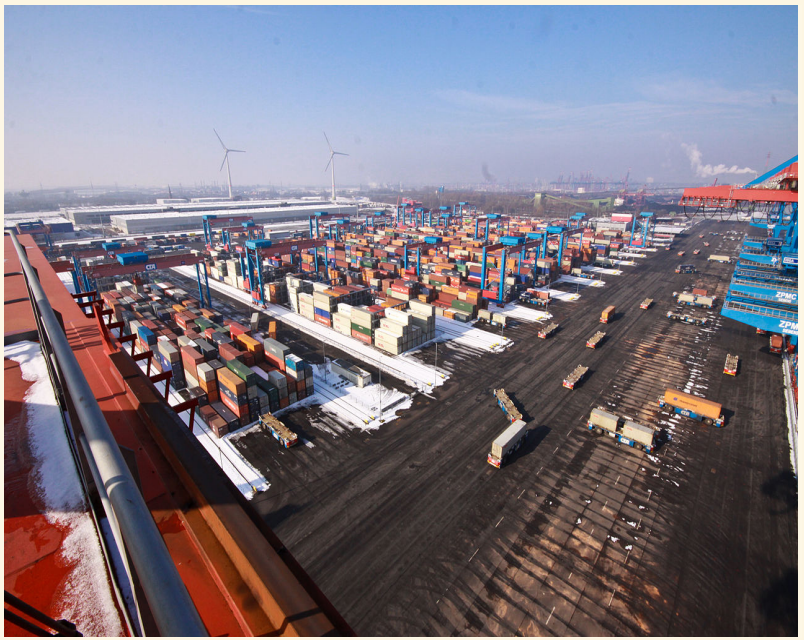
... sind Sie in der Lage die Docker Kommandozeile zu verwenden.

... können Sie eigene Docker Images und Container erzeugen.

WAS IST EIN CONTAINER?

„CONTAINERIZATION“ IN DEN 60ER JAHREN

- Bessere Platznutzung (genormte Maße, stapelbar)
- Höhere Transportsicherheit (Beschädigung, Diebstahl)
- Effizientere Be- und Entladung sowie Transport
- Geschlossene Transportkette (Land, See)

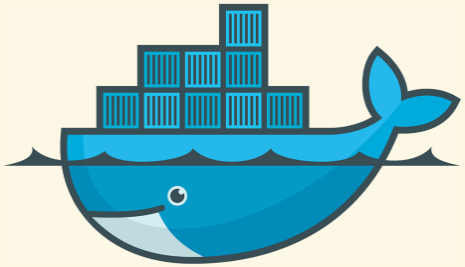


WAS IST EIN DOCKER CONTAINER?

Ein Container ist ein **standardisiertes** Paket, das die Softwareanwendungen mit allen Abhängigkeiten (Laufzeiten, System Tools, Bibliotheken, ..) **portabel** und **isoliert** verpackt.

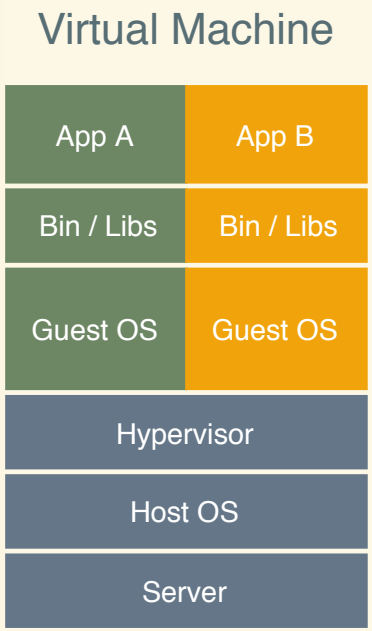
Dadurch können Anwendungen schnell, einfach und **reproduzierbar** auf unterschiedlichen Systemen ausgeführt werden.

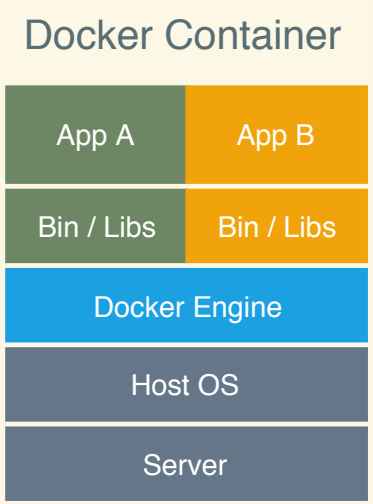
„CONTAINERIZATION“ IN DEN 2010ER JAHREN



Transportcontainer	Docker Container
"Bessere Platznutzung"	Platzersparnis durch Verzicht auf ein komplettes Betriebssystem
"Höhere Transportsicherheit"	Isolation des Containers zu anderen Prozessen
"Effizientere Be- und Entladung sowie Transport"	Standardisierter Deploymentprozess unterstützt durch effiziente Tools
"Geschlossene Transportkette"	Gewisse Unabhängigkeit von Betriebssystem & Cloudanbieter

VIRTUELLE MASCHINE VS. DOCKER CONTAINER





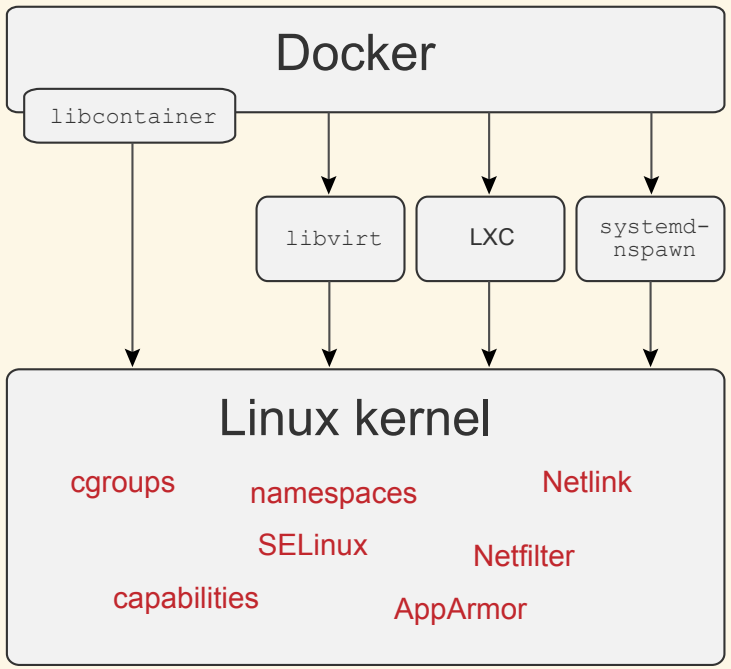
	VM	Container
Grundlage:	Hardware Virtualisierung	Isolation (Prozesse, Dateisystem, Rechte, ..)
Gastsystem:	eigener Kernel	Kernel des Hostsystems wird mitverwendet

DOCKER ISOLATION

Docker nutzt Features des Linuxkernels zur Isolation:

- **cgroups**: Allokation von Rechenleistung und Bandbreite (zB. CPU, RAM, Datei- und Netzwerkzugriff)
- **namespaces**: Isolation von Prozessen, Nutzern, Files, usw.

und weitere ...



VORTEILE VON CONTAINERTECHNOLOGIE

Containerbasierte Anwendungen haben ...

- ... ein **identisches Verhalten** auf unterschiedlichen Systemen.
- ... einen **geringeren Ressourcenverbrauch** gegenüber VMs (Speicherplatz, RAM, CPU).
- ... eine **signifikant schnellere Startzeit** gegenüber VMs.

DEMO: CONTAINER STARTEN

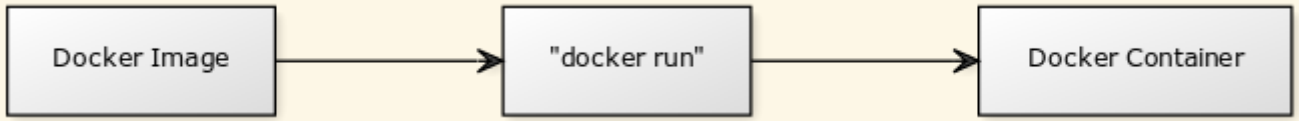
```
docker run hello-world
```

DOCKER BEFEHLE: CONTAINER

Befehl	Beschreibung
`docker run <image>`	Startet einen neuen Container auf Basis des angegebenen Docker Images. Unterstützt weitere Parameter z.B. Portfreigabe `docker run -d -p 80:8000 crccheck/hello-world`.
`docker ps [-a]`	Zeigt alle laufenden Container (-a auch die bereits gestoppten)
`docker start <container>`	Startet einen bereits existierenden/gestoppten Container
`docker stop <container>`	Stoppt einen laufenden Container

Befehl	Beschreibung
`docker rm <container>`	Entfernt einen Container

DOCKER IMAGES & CONTAINER



Docker Container	≙	gestartete Instanz eines Images
Docker Image	≙	Bauplan bzw. Datenbasis eines Containers

ÜBUNG 1: DOCKER BASICS

DOCKER REGISTRY

Docker Images werden in einer Registry verwaltet:

- Standardmäßig verbindet sich ihr Docker Client mit der **offiziellen Registry**: **http://hub.docker.com**
- Es ist allerdings auch möglich eine **eigene Registry** zu betreiben: registry image.
- .. und es existieren verschiedene *komerzielle Angebote*:
  - Google Cloud Container Registry
  - Amazon Elastic Container Registry

DOCKER IMAGES

Ein Docker Image wird durch ein sog. **Dockerfile** definiert.

Beispiel:

```
FROM alpine:latest

COPY ./hello.sh /hello.sh

RUN chmod 755 /hello.sh

CMD ["sh", "hello.sh"]
```

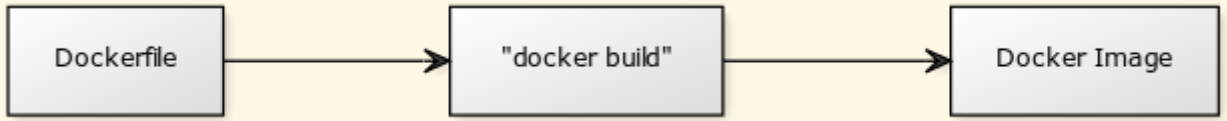
- **FROM** Beschreibt das Basisimage auf das aufgebaut wird.
- **COPY** Kopiert eine File/einen Ordner in das Image.
- **RUN** Führt einen Befehl innerhalb des Images während des Bauvorgangs aus.
- **CMD** Beschreibt den Standard-Einstiegspunkt in die Anwendung.

DEMO DOCKER IMAGE

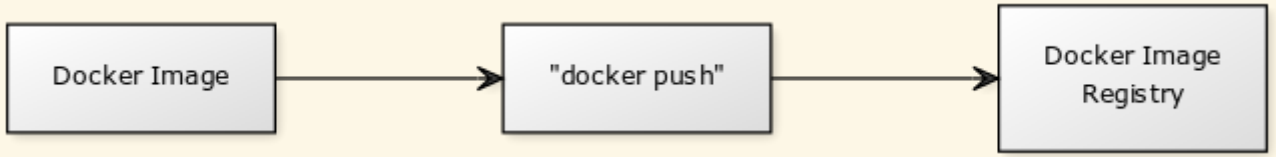
DOCKER IMAGES #2

- Bauen des Image

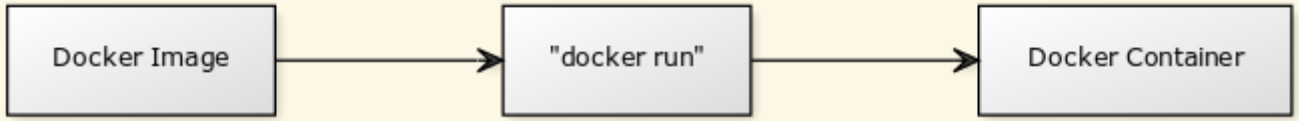
```
docker build . -t myimage
```



- Optional: Publizieren auf der Registry



- Ausführen als Containers



ÜBUNG 2: DOCKER IMAGES

DOCKER PORTFREIGABEN

- Docker wird in der modernen Cloudentwicklung eingesetzt um **Webanwendung** (*"Microservices"*) als **Container** zu paketieren.
- Aus Sicherheitsgründen werden **per default keine Ports** aus den Containern an das Host System **freigegeben**.
- Portfreigaben müssen **explizit** beim `run` Befehl angegeben werden:

```
docker run -p HOST_PORT:CONTAINER_PORT`
```

- Demo:

```
# -d steht für *detached* bzw. Ausführung im Hintergrund
docker run -d -p 80:80 rancher/hello-world
```

DOCKER VOLUMES

- Beim entfernen eines Containers gehen alle Daten in diesem verloren.
- Das ist natürlich keine Option für Anwendungen die *stateful* sind. Beispiel: Datenbanksysteme.
- Für solche Szenarien gibt es Docker Volumes. Mit diesen können Dateipfade des Hostsystems in den Containers eingehängt werden. Somit überdauern diese den Containerlifecycle.
- Demo:

```
docker run -d -v /my/folder:/web -p 8080:8080 halverneus/static-file-server:latest
```

ÜBUNG 3: DOCKER IMAGES - PORTS & VOLUMES

ZUSAMMENFASSUNG & AUSBLICK

- Docker Container sind leichtgewichtiger als virtuelle Maschinen, bringen trotzdem viele ihrer Vorteile mit.
- Docker Container sind die Basis der *modernen Anwendungsentwicklung* in der *Cloud*.
- Um Serverressourcen möglichst effizient zu nutzen sollten Container entsprechend der Auslastung möglichst effizient auf einem Servercluster verteilt werden. sog. *Container-Orchestration-Systeme* wie Kubernetes, Mesos, Docker Swarm helfen dabei.

END